



APPLICATION PROGRAMMING MACROS

HCC/MVS 3.0

HOST

COMMUNICATION

CONTROL

1st edition



IMPRINT

1st Edition

This Manual has been written with utmost care. Textual or formal errors still cannot be excluded!

Protected trademarks are not marked as such in this Manual. The fact that these trademarks are not shown does not imply that the trade names are free for use.

All rights withheld, including those arising from applications for proprietary rights. The publisher retains all rights of disposition, such as copying or distribution.

This Manual is for internal use only.

Subject to changes without notice.

Publisher:

EMASS/GRAU Software GmbH,
Gottlieb-Daimler-Straße 17/3, 74385 D-Pleidelsheim

1st Edition in July 1995

CONTENTS

1 APPLICATION PROGRAMMING MACROS FOR HCC/MVS 3.0.0.....	1-1
1.1 ASSOCIATED DOCUMENTS	1-1
2 MACRO DESCRIPTION.....	2-1
2.1 ZHCALL	2-1
2.1.1 GENERAL INFORMATION	2-1
2.1.2 SYNTAX.....	2-1
2.1.3 PARAMETERS	2-1
2.1.4 PASSING PARAMETERS	2-1
EXAMPLE 1.....	2-2
2.1.5 EXAMPLE 2.....	2-2
2.1.6 EXAMPLE 3.....	2-3
2.2 ZHCAM.....	2-4
2.2.1 GENERAL INFORMATION	2-4
2.2.2 SYNTAX.....	2-4
2.2.3 PARAMETERS	2-4
2.2.4 EXAMPLE	2-4
2.3 ZHCARCH.....	2-5
2.3.1 GENERAL INFORMATION	2-5
2.3.2 SYNTAX.....	2-5
2.3.3 PARAMETERS	2-5
2.3.4 EXAMPLE 1.....	2-6
2.3.5 EXAMPLE 2.....	2-6
2.4 ZHCBEGL.....	2-7
2.4.1 GENERAL INFORMATION	2-7
2.4.2 SYNTAX.....	2-7
2.4.3 PARAMETERS:	2-8
2.4.4 EXAMPLE1	2-9
2.4.5 EXAMPLE2	2-9
2.5 ZHCCALL	2-10
2.5.1 GENERAL INFORMATION	2-10
2.5.2 SYNTAX.....	2-10
2.5.3 PARAMETERS	2-10
2.5.4 EXAMPLE	2-10
2.6 ZHCDMP	2-11
2.6.1 GENERAL INFORMATION	2-11
2.6.2 SYNTAX.....	2-11
2.6.3 PARAMETERS:	2-11
2.6.4 EXAMPLE	2-11
2.7 ZHCDOM.....	2-12
2.7.1 GENERAL INFORMATION	2-12
2.7.2 SYNTAX.....	2-12
2.7.3 PARAMETERS:	2-12
2.7.4 EXAMPLE	2-12
2.8 ZHCEXIT.....	2-13

2.8.1	GENERAL INFORMATION	2-13
2.8.2	SYNTAX.....	2-13
2.8.3	PARAMETERS	2-13
2.8.4	EXAMPLE	2-13
2.9	FUNCTION.....	2-14
2.9.1	CONCEPT.....	2-14
2.10	ZHCFUNC.....	2-16
2.10.1	GENERAL INFORMATION	2-16
2.10.2	SYNTAX.....	2-16
2.10.3	PARAMETERS	2-17
2.10.4	EXAMPLE 1.....	2-19
2.10.5	EXAMPLE 2.....	2-19
2.10.6	EXAMPLE 3.....	2-20
2.11	ZHCGHT	2-21
2.11.1	GENERAL INFORMATION	2-21
2.11.2	SYNTAX.....	2-21
2.11.3	PARAMETERS	2-21
2.11.4	EXAMPLE	2-21
2.12	ZHCISP.....	2-22
2.12.1	GENERAL INFORMATION	2-22
2.12.2	SYNTAX.....	2-22
2.12.3	PARAMETERS	2-22
2.12.4	EXAMPLE	2-22
2.13	ZHCMGN	2-23
2.13.1	GENERAL INFORMATION	2-23
2.13.2	SYNTAX.....	2-23
2.13.3	PARAMETERS	2-24
2.13.4	EXAMPLE 1.....	2-25
2.13.5	EXAMPLE 2.....	2-25
2.14	ZHCMSG	2-26
2.14.1	GENERAL INFORMATION	2-26
2.14.2	SYNTAX.....	2-26
2.14.3	PARAMETERS	2-26
2.14.4	EXAMPLE	2-27
2.15	ZHCSUE	2-28
2.15.1	GENERAL INFORMATION	2-28
2.15.2	SYNTAX.....	2-28
2.15.3	PARAMETERS	2-28
2.15.4	EXAMPLE	2-29
2.16	ZHCSUX	2-30
2.16.1	GENERAL INFORMATION	2-30
2.16.2	SYNTAX.....	2-30
2.16.3	PARAMETERS	2-30
2.16.4	EXAMPLE	2-31
2.17	ZHCTME	2-32
2.17.1	GENERAL INFORMATION	2-32
2.17.2	SYNTAX.....	2-32
2.17.3	PARAMETERS	2-32
2.17.4	EXAMPLE 1.....	2-33
2.17.5	EXAMPLE 2.....	2-33
2.18	ZHCTRE.....	2-34
2.18.1	GENERAL INFORMATION	2-34



2.18.2 SYNTAX.....	2-34
2.18.3 PARAMETERS	2-34
2.18.4 EXAMPLE	2-35
2.19 ZHCUPE.....	2-36
2.19.1 GENERAL INFORMATION	2-36
2.19.2 SYNTAX.....	2-36
2.19.3 PARAMETERS	2-36
2.20 ZHCUPX.....	2-37
2.20.1 GENERAL INFORMATION	2-37
2.20.2 SYNTAX.....	2-37
2.20.3 EXAMPLE FOR ZHCUPE/ZHCUPX.....	2-37



APPLICATION PROGRAMMING MACROS FOR HCC/MVS 3.0

1 APPLICATION PROGRAMMING MACROS FOR HCC/MVS 3.0

1.1 ASSOCIATED DOCUMENTS

Manual	Order number German	Order number English	Reference
General Information Manual	N/A	600226-A	G
Installation/Customization Guide	N/A	600227-A	G
System Reference Guide	N/A	600233-A	G
Operators Guide	N/A	600231-A	G
Command Reference	N/A	600223-A	G
Conversion Notebook	N/A	600224-A	G
Installation/Customization Reference	N/A	600228-A	G
Messages and Codes	N/A	600230-A	G
Operator Quick Reference	N/A	N/A	N/A
ISPF User Guide	N/A	600229-A	G
Release Guide	N/A	600232-A	G
Diagnostic Guide	N/A	600225-A	I
Application Programming Macros	N/A	600222-A	I

G This Manual is part of the standard HCC documentation. Further copies of the Manuals can be obtained from the publisher.

I This Manual is only intended for internal use within GRAU Software GmbH and EMASS/GRAU Storage Systems. This manual is not part of the standard HCC documentation and cannot be purchased as part of the additional documentation



APPLICATION PROGRAMMING MACROS FOR HCC/MVS 3.0

2 MACRO DESCRIPTION

2.1 ZHCALL

2.1.1 General information

ZHCALL serves to call subprograms in a uniform manner. ZHCCALL should be used whenever possible.

2.1.2 Syntax

ZHCALL

ZHCALL must be followed by one or more blanks

MOD=modname

modname = max. 8 character string

, **PARM=parm**

parm = RX type address or (0) - (14)

, **PARM=(parm1,parm2,...)**

2.1.3 Parameters

MOD=MODNAME

Module name (CSECT) of the program to be called.

PARM=parm / (parm1,parm2,...)

Passes parameters to the called program (see **Passing parameters**).

2.1.4 Passing parameters

Parameters can be passed to the called program in two ways:

Method 1:

LA R1,PARMAREA

ZHCALL MOD=UPRO

R1 points to a storage area containing all the data required by the UPRO.

Method 2:

ZHCALL MOD=UPRO,PARM=PARMAREA

R1 points to a storage area containing the addresses of the parameters specified in PARM=

Example 1

Program HPRO calls program UPRO and passes the parameters as a **list** in the macro:

```

HPRO:
      ZHCALL MOD=UPRO, PARM=UPROPARM

      UPRO      DC      V(UPRO1)
      UPROPARM  DS      0F
      PARM1     DC      CL8'Testparm'
      PARM2     DC      A(anyaddr)

      UPRO:
          entry-code
          L      R1,0(R1)      pointer to field UPROPARM
          *                           in HPRO
          MVC    UPARM(8),0(R1) move parm1
          MVC    UADDR(4),8(R1) move parm2

```

2.1.5 Example 2

Program HPRO calls program UPRO and passes the parameters as **single items** in the macro.

```

HPRO:
      ZHCALL MOD=UPRO, PARM=( PARM1 , PARM2 )

      UPRO      DC      V(UPRO1)
      UPROPARM  DS      0F
      PARM1     DC      CL8'Testparm'
      PARM2     DC      A(anyaddr)

      UPRO:
          entry-code
          L      R1,0(R1)      pointer to 1st.parm
          *                           in HPRO
          L      R2,0(R1)
          MVC    UPARM(8),0(R2) move parm1
          L      R2,4(R1)
          MVC    UADDR(4),0(R2) move parm2

```

MACRO DESCRIPTION

2.1.6 Example 3

Program HPRO calls program UPRO and passes the parameters EXTERNAL using R1.

```
HPRO:  
      LA      R1 ,UPROPARM  
      ZHCALL MOD=UPRO
```

```
UPRO   DC     V(UPRO1)  
UPROPARM DS     0F  
PARM1   DC     CL8 'Testparm'  
PARM2   DC     A(anyaddr)
```

```
UPRO:  
      entry-code  
      MVC    UPARM(8),0(R1) move parm1  
      MVC    UADDR(4),8(R1) move parm2
```

2.2 ZHCAM

2.2.1 General information

ZHCAM serves to set the correct address mode in a program. When calling a program with an instruction that assumes the AMODE of the calling program, save the original address mode before using ZHCAM (see example)

2.2.2 Syntax

ZHCAM

ZHCAM must be followed by one or more blanks

AMODE=24
AMODE=31

2.2.3 Parameters

AMODE=24 / 31 Sets the program to the specified address mode

2.2.4 Example

A program with AMODE31 / RMODE24 is called by both 24 and 31 bit programs per BALR. This program must first determine the original address mode and then read a storage area above 16 MB. Reset the original AMODE afterwards.

```
*----- Check AMODE of program -----
    XR   R1,R1
    BSM  R1,0      set bit 0 of R1 to 1 when AMODE=31
    ST   R1,SAVEAM
*----- Now read the values above 16 MB -----
    TM   SAVEAM,X'80'    check for 31 bit mode
    BO   READAREA
*----- Caller was in 24 bit mode, so
*----- 31 bit mode must be set
    ZHCAM AMODE=31
    READAREA DS   0H
    .
    .
*----- This 31 bit operation is now completed,
*     check the original AMODE again
    TM   SAVEAM,X'80'    check for 31 bit mode
    BO   CONTINUE
    ZHCAM AMODE=24
    CONTINUE DS   0H
    .
    .
    SAVEAM DC   F'0'
```

2.3 ZHCARCH

2.3.1 General information

ZHCARCH simplifies complex archive operations. Parameters are passed to SU8200 and then passed as a function to ZHC05900 with the FADD service. The parameters passed are analyzed there and processed accordingly.

2.3.2 Syntax

ZHCARCH

ZHCARCH must be followed by one or more blanks

RTYP=1	Freevol for OD's
RTYP=21	Arch. operations after LJB/UJB or UPV
ACT=UPD	
PARM=	A type address or (2) - (12)
FUS=NEW	
FUS=OLD	
WORK=	A type address or (1) - (12)

2.3.3 Parameters

ACT=UPD

Action to be performed. UPD is the only possible action at the present time.

FUS=OLD/NEW

FUS=OLD specifies that the currently active function should be passed to ZHC05900 (for example, the FREEVOL function).

FUS=NEW specifies that a new function should be created with the FCOPY service and then passed to ZHC05900.

WORK=

Work area address for re-entrant programs.

RTYPE=1 , PARM=

Transfers the data from the FREEVOL command to the Archive. The volsers in one or more buffers are passed to ZHC05900. Here, all OD "NOTFULL" bits are first determined and then set according to the new data. The parameter area for ZHC05900 must be 2 fullwords in size.

The first fullword must contain the number of parameters being passed (1), and the second fullword the address of the first or only buffer with selected volsers.

RTYPE=21 , PARM=

This RTYP triggers Archive updates with the following operations:

LJB positive / UJB positive / UPV between status OE/OB <--> OJ .

ZHC05900 performs the following updates:

A volser/B volser on disk,

JB control/volser record on disk,

JB control/volser entry memory

A/B volser memory

The parameter area for ZHC05900 must be 3 fullwords in size.

The first fullword must contain the number of parameters being passed (2), the second fullword the corresponding JBTAB volser entry and the third fullword the address of the JBTAB control entry.

2.3.4 Example 1

A FREEVOL,,TYPE=OPT command has read the sequential file and stored all the selected volsers in buffers. H59WORKA contains the address of the first buffer. PARMAREA is the parameter area name, ARCWORK is the re-entrant work area name. The currently active function (FREEVOL) is to be passed to ZHC05900. The parameter area data are stored in the function entry of the FREEVOL function where ZHC05900 can retrieve and evaluate the data.

```

LA      RF,1           ind.: 1 parameter follows
ST      RF,PARMAREA    save in 1st word of parmarea
L       RF,H59WORKA   get pointer to volser buffer
ST      RF,PARMAREA+4  save in 2nd word of parmarea
ZHCARCH RTYP=1,PARM=PARMAREA,WORK=ARCWORK,FUS=OLD
LTR     RF,RF

```

2.3.5 Example 2

A UJB command has been successfully executed. The corresponding changes must now be made in the Archive. The corresponding JBTAB volser or control entries can be determined based on the VOLSERs and passed as parameters to ZHC05900. The original function (UJB) should NOT be used (FUS=NEW).

```

XC      B13PARM,B13PARM  Clear parmarea
LA      RF,2           ind.: 2 parameters follow
ST      RF,B13PARM     save in 1st word of parmarea
MVC    B13PARM+4,B13JBV move JBV pointer to 2nd word
MVC    B13PARM+8,B13JBC move JBC pointer to 3rd word
ZHCARCH RTYP=21,ACT=UPD,PARM=B13PARM,FUS=NEW
LTR     RF,RF

```

2.4 ZHCBEG

2.4.1 General information

ZHCBEG was developed to simplify ENTRY/EXIT coding in programs. The macro serves to specify all the parameters required for ENTRY/EXIT coding. A LABEL is provided for the EXIT: ENDxxxx where xxxx are the first four characters of the CSCT parameter.

This macro can also be used to start re-entrant programs. The parameters WORKNAME, WORKREG and WORKLEN are used in this case and **all** these parameters must be specified.

2.4.2 Syntax

ZHCBEG

ZHCBEG must be followed by one or more blanks

CSCT=csectname

csectname=character string

, AM=31
, AM=24

Default: AM=31

, RM=24
, RM=31
, RM=ANY

Default: RM=ANY

, BR=basereg
, BR=(basereg,basereg,...)

basereg=any valid register number

Default: BR=12

, EQU=NO
, EQU=YES

Default: EQU=NO

, PRINT=GEN
, PRINT=NOGEN

Default: PRINT=NOGEN

, TYPE=EXT
, TYPE=INT

Default: TYPE=EXT

, VERS=3.0.000

7 byte character string

, WORKNAME=workname
, WORKREG=regnum
, WORKLEN=lengthname

workname = character string
regnum = any valid register number
lengthname = name of the EQUATE specifying the length of WORKNAME

2.4.3 Parameters:

CSCT=csectname

Name of the CSECT valid for ZHCBEG. This is usually the program name as well. This parameter is mandatory.

AM=24/31

ADDRESS MODE of the program.

RM=24/31/ANY

RESIDENCY MODE of the program.

BR=basereg**BR=(basereg,basereg,...)**

Base register(s) of the program. Default BR=12 is assumed when this parameter is not specified.

EQU=NO/YES

Specifies whether register equates should be generated or not. Default is EQU=NO. R0-RF equates have already been generated in HCC in the ICT.

PRINT=GEN/NOGEN

Specifies whether a macro resolution should appear in the compilation list or not. Default is PRINT=NOGEN.

TYPE=INT/EXT

Specifies whether the CSECT is an internal or external CSECT. For TYPE=EXT, the end label is created from the first three characters of the CSECT name plus the word "END". For TYPE=INT, the first five characters plus the word "END" are used.

ZHCBEG ...TYPE=INT can be used instead of ZHCUPE/ZHCSUE.

The following parameters **MUST** be specified when the program is **RE-ENTRANT**.

WORKNAME=workname

Name of a DSECT to be used as re-entrant work area. This DSECT must be specified in the programs involved.

WORKREG=regnum

Base register for the work area WORKNAME.

WORKLEN=length

Work area length (see example)

MACRO DESCRIPTION

2.4.4 Example1

A non-re-entrant program with the names TESTPROG, AMODE 31,RMODE 24, base registers 11,12 and EQUATES creation.

```

ZHCBEG CSCT=TESTPROG,AM=31,RM=24,BR=(11,12),      *
          EQU=YES

.

.

Any program code

.

.

L    RF,RETCODE           Load retcode to R15
B    ENDTST                Return
FELD1 DS F
RETCODE DS F

```

2.4.5 Example2

A re-entrant program with the names RENTPROG, AMODE 31,RMODE 24, base registers 11,12

```

ZHCBEG CSCT=RENTPROG,AM=31,RM=24,BR=(11,12),      *
          WORKNAME=WORKA,WORKREG=10,WORKLEN=WLEN

.

.

Any program code

.

.

L    15,RETCODE           Load retcode to R15
B    ENDRENT                Return
WORKA DSECT
FELD1 DS F
RETCODE DS F
WLEN   EQU *-WORKA

```

2.5 ZHCCALL

2.5.1 General information

ZHCCALL serves to call programs in a uniform manner. The program EPAs should be entered in ZHCEPA. This macro must be present as mapping macro in every program using ZHCCALL. The following routines create EPAs in ZHCEPA: ZHC00101 ("Module Loader"), ZHC06800 ("Re-entrant Subroutines Module"). This macro passes control to the called routine with BASSM. It is recommended to include the **ZHCBEG** (Entry/Exit code) macro in the called routine because it uses a BSM instruction to pass control.

2.5.2 Syntax

ZHCCALL

ZHCCALL must be followed by one or more blanks

EP

RX type address

EPA

(2) - (12)

, PARM=parm

parm = RX type address or (1) - (14)

2.5.3 Parameters

EP=

Field name from ZHCEPA must be specified.

EPA=

EPA of the called routine in a register must be specified.

PARM=

Address of a parameter area can be entered. The parameters for the called program can also be passed directly via R1.

2.5.4 Example

To call a program (ZHC04903).

```
*----- Call ZHC04903 -----
LA   R1 , PARMAREA
ZHCCALL EP=EPA04903    call the OD Flip routine
LTR  RF ,RF
```

2.6 ZHCDMP

2.6.1 General information

ZHCDMP supports displaying any hexadecimal values from any point in the program on the console. An address within the HCC must be specified (FROM=), together with a length (L=) and a header text to be shown at the start of the dump (HDR=). The calling program must provide a 100 byte work area called WORK as well as the ZHCXCV macro as data area otherwise assembly errors will occur.

2.6.2 Syntax

ZHCDMP

ZHCDMP must be followed by one or more blanks

FROM=stor addr ,

stor addr = RX type address or Reg. (2) - (12)

L=length-value ,

length-value = dec. number or Reg. (2) - (12)

HDR= 'text'

text = max. 8 byte text string

2.6.3 Parameters:

FROM=stor addr

Address of a program area to be "unhexed". The address can be located in register 2-12 or in a fullword.

L=length-value

Length to be "unhexed" as from FROM=. The length can be specified as a length in the parameter or in register 2-12.

HDR='text'

8 byte text to be shown before the actual dump data.

2.6.4 Example

Permanently display a 40 byte control block CB1 in a program. This control block contains arbitrary hexadecimal values.

```
ZHCDMP FROM=CB1,L=40,HDR=' PRNT CB1 '
.
.
.
WORK    DS      CL100
CB1     DC      X180'FF'
ZHCXCV
```

2.7 ZHCDOM

2.7.1 General information

ZHCDOM serves to delete action messages from the console.

2.7.2 Syntax

```
ZHCDOM <parm-id1=parameters1>[ ,parm-id2=parameters2,... ]
```

2.7.3 Parameters:

ID=	POINTER TO WTO ID
MSG=	POINTER TO MESSAGE ID
VOL=	POINTER TO VOLSER
SYS=	POINTER TO SYSTEM NO
ROB=	POINTER TO ROBOT NO
DEV=	POINTER TO DEVICE NO / AML UNIT
COOR=	POINTER TO COORDINATE
JOB=	POINTER TO JOBNAME
HID=	POINTER TO HCC ID
AID=	POINTER TO AMU ID
SQNR=	POINTER TO AML SEQUENCE NO
RETC=	POINTER TO AML RETURN CODE
WORK=	POINTER TO WORK AREA FOR REENTRANCE
ALL=	'ALL' TO DELETE ALL MESSAGES

At least 1 parameter must be specified. The sequence is optional.

2.7.4 Example

Delete message HAC332 after the pending mount has been executed with an FMA/FMM command.

```
HACDOM MSG=MSG332 , DEV=DAUNIT , JOB=DMJOB
```

2.8 ZHCEXIT

2.8.1 General information

ZHCEXIT serves to terminate programs in cooperation with the ZHCBEG. This macro uses the End label generated in ZHCBEG.

2.8.2 Syntax

ZHCEXIT

ZHCEXIT must be followed by one or more blanks

RC=

Return code: decimal value or (0) - (15)

PRINT=GEN
PRINT=NOGEN

Default: PRINT=NOGEN

2.8.3 Parameters

RC=

Provides the calling program with the return code in register 15.

PRINT=GEN/NOGEN

Specifies whether the macro resolution should be displayed (PRINT=GEN) or not (PRINT=NOGEN).

2.8.4 Example

The program ZHC04500 :

```

ZHCBEGL CSCT=ZHC04500,AM=31,RM=ANY,                               *
      WORKNAME=WORK,WORKREG=10,WORKLEN=WLEN
.
.
.
*-----*
*      Exit Error          *
*-----*
X0900000 DS      0H
      ZHCEXIT RC=4
*-----*
*      Exit Normal         *
*-----*
Z0010000 DS      0H
      ZHCEXIT RC=0
*-----*
WTOID     DC      F' 0 '

```

2.9 FUNCTION

2.9.1 Concept

Function is a closed unit represented by an entry in the Function table. The following are included at the moment:

- command to be executed (for example: VI I01)
- source (for example: ABBASEND)
- triggered by (for example: Console ID)
-

Function is assigned to an active TASK/SUBTASK using a further table, the PGMTAB. An entry is made in this table for each subtask started where its address is set in the TCB of the new TASK in the TCBUSER field. At the same time, a so-called "DEFAULT function" is created for the task when the subtask starts, and linked to the corresponding PGMTAB entry using the PGMFUS/PGMFUS1 field. (TINIT Service)

A further Service is called (TTERM-Service) when the subtask terminates to release the PGM entry, the default function entry and a possible active function entry. The TCBUSER field in the SUBTASK-TCB is also cleared.

When a command is received, a function entry is created and marked as active in the corresponding PGM entry. A check is also made as to whether an ABS output buffer is required. (FINIT Service)

The previous entry is released when the function ends and the default function in the corresponding PGM entry becomes the active function. (FTERM-Service)

When a function receives an Archive/robot action, it first terminates when the ROB RESPONSE is received. After entry in the respective SYS/ROB queue (SU2300), the data queue pointer is entered in the corresponding function entry and the relative position of the function entry in the data queue entry. The default function in the respective PGM entry becomes the active function. (FRESET-Service).

When the ROB RESPONSE is received, the corresponding function is determined from the data queue entry (from the relative position) (FREST-Service) and entered as active function in the corresponding PGM entry. (FSET Service).

When robot error codes trigger an entry in the ERROR queue, the source of the original command is set to "CONSOLE" and the function marked as "HELD".

A new function entry is generated for the SCH P response for an insertion. This entry has "INTERNAL" as source, all output goes to the console irrespective of the source of the triggering **VI** command (FCOPY-Service). This new function entry is entered as active function in the PGM entry (FSET-Service).

The respective SYS/ROB queue is linked to the active function as soon as the new data queue entry is made. (SU2300). The original function (**VI**) is then entered as active function in the PGM entry. (FSET-Service).

The FGET Service is a general service to be seen as independent of any functionality. This Service acquires the following information:

- pointer to active function entry
- pointer to default function entry
- pointer to current PGM entry



MACRO DESCRIPTION

This function entry must be passed to the corresponding subtask when a function that triggers any sort of work in the subtask is now started. The subtask then switches the function entry from the PGM entry of the triggering function, to its own PGM entry and enters it as the active function. The default function is also set as the active function in the PGM entry of the triggering function. (FSWI-Service). The FTERM Service is called when the function ends.

All these function services are called using the **ZHCFUNC** macro.

2.10 ZHCFUNC

2.10.1 General information

Refer to the FUNCTION Section.

2.10.2 Syntax

ZHCFUNC

ZHCFUNC must be followed by one or more blanks

FUNC=TINIT
 FUNC=TERMINATE
 FUNC=TCANC
 FUNC=FINIT
 FUNC=TERMINATE
 FUNC=FGET
 FUNC=FSWI
 FUNC=FSET
 FUNC=FREST
 FUNC=FRESET
 FUNC=FCOPY
 FUNC=FADD

,ICT=stor addr

stor addr = RX type addr or Reg. 2-12

Default: ICT=(6)

,WORK=stor addr

stor addr = RX type addr or Reg 2-12

Default: NONE

,MOD=stor addr

stor addr=RX type addr or Reg. 2-12

Note: required FOR FUNC=TINIT,
FUNC=FADD

,CMS=stor addr

stor addr=RX type addr or Reg. 2-12

Note: required FOR FUNC=FINIT

Default: CMS=NONE

,FUS=stor addr

stor addr = RX type addr or Reg. 2-12

Note: required FOR FUNC=FSET
FUNC=FCOPY
FUNC=FSWI
FUNC=FADD

, OLDPGM=stor addr

stor addr = RX type addr or Reg. 2-12

Note: required FOR FUNC=FSWI
required FOR FUNC=FSET

,NEWPGM=stor addr

stor addr=RX type addr or Reg. 2-12

Note: required FOR FUNC=FSWI
offset = RX type addr or Reg. 2-12

,OFFS=offset

Note: required FOR FUNC=FREST
offset = RX type addr or Reg. 2-12

,HACCCMD=stor addr

Note: required FOR FUNC=FCOPY

2.10.3 Parameters

WORK=storaddr

WORK= parameter specifies where the parameter list for the function service module ZHC04800 is located. This parameter **must** be specified for re-entrant programs.

The parameter list is created inline when omitted for NON-re-entrant programs. In such cases, R1 contains the address of the INLINE parameter list when the caller is returned to. Any Service results are returned in the parameter list. This parameter is valid for all function calls.

ICT=storaddr

Supports sending the HCC-ICT address to the function service module. R6 is assumed as default. This parameter is valid for all function calls.

FUNC=TINIT

Initializes a PGM entry and a DFLT. function entry during subtask start. Links the new PGM entry to the active TCB.

,MOD=modname

Name of the first module of a subtask.

Output:

Word 4 of the parameter list contains a pointer to the PGM entry.

FUNC=TERMINATE

Releases the PGM entry, dflt. function entry and active function entry. Used at the end of a subtask.

FUNC=FINIT

Creates a new function based on a command and enters the new function as active function in the corresponding PGM entry.

,CMS=storaddr / NONE

Points to an entry in HCC's COMMANDSTACK.

Output:

Word 3 of the parameter list contains a pointer to the new function entry.

FUNC=FTERM

Releases the active function entry when the function ends and enters the dflt. function as active function in the PGM entry.

FUNC=FGET

Returns pointers to the active function, default function and current PGM entry to the caller.

Output:

Word 3 of the parameter list contains a pointer to the active function entry

Word 4 of the parameter list contains a pointer to the default function entry

Word 5 of the parameter list contains a pointer to the current PGM entry

FUNC=FSWI

Switches a function entry from one ("OLDPGM") PGM entry to another ("NEWPGM").

FUS=storaddr

Points to the function entry in question

OLDPGM=storaddr

Points to the original PGM entry

NEWPGM=storaddr

Points to the new PGM entry

FUNC=FSET

Sets the function entry specified in FUS= as active function in the PGM entry specified in OLDPGM=.

FUS=storaddr

Specifies the function entry to become the active function.

OLDPGM=storaddr

Specifies the PGM entry in which the function entry is to be entered as the active function.

FUNC=FREST

Uses the relative offset in the Function table to calculate the address of a function entry.

OFFS=storaddr

The relative offset can be specified in a register, or must be in a **HALFWORD** (e.g.: QADFUS in ZHCDQR).

Output:

Word 3 of the parameter list contains a pointer to the function entry determined with FREST.

Word 4 of the parameter list contains a pointer to the current PGM entry.

Example:

```

ZHC FUNC FUNC=FGET,WORK=FUNCPRM
L RF,FUNCPRM+8           get pointer to active function entry
ST RF,FUSOLD
L RF,FUNCPRM+16          get pointer to current PGM entry
ST RF,PGMCURR
ZHC FUNC FUNC=FREST,OFFS=QADFUS,WORK=FUNCPRM
L R2,QADFUS
ZHC FUNC FUNC=FREST,OFFS=(R2),WORK=FUNCPRM
L RF,FUNCPRM+8           get pointer to found function entry
ST RF,FUSNEW
ZHC FUNC FUNC=FSET,OLDPGM=PGMCURR,FUS=FUSNEW, *
                  WORK=FUNCPRM

```

OR
FUNC=FRESET

Sets the default function entry as active function in the current PGM entry.

FUNC=FCOPY

Creates a new function entry. It is used as a model for the function entry specified in FUS=.

FUS=storaddr

Points to the function entry to be used as model for the new function entry. However, the internal flag is used as ORIGIN and the character string from HACCCMD (8 bytes) as FUSMSG.

HACCCMD=storaddr

Points to a field containing the mnemonic term for the new function.

Output:

Word 3 of the parameter list contains a pointer to the new function entry.

FUNC=FADD

Creates a new entry in the Function_pointer_queue of the subtask, specified in the macro with MOD=.

2.10.4 Example 1

Create a new function for ZHC06000.

```
ZHCFUNC FUNC=TINIT,MOD=MD060,WORK=FUNCPRM
.
.
.
FUNCPRM DS 5F
```

2.10.5 Example 2

Start the LOGWRITER task ZHC060 using the **LOGSTART** command:

```
ZHCFUNC FUNC=TINIT,WORK=FUNCPRM,MOD=MD060
L RF,FUNCPRM+12      Get pointer to new
                      PGM entry
ST RF,PGMCURR        Save it
* Pointer to LOGSTART function entry passed as
* a parameter in ATTACH
.
.
.
ZHCFUNC FUNC=FSWI,FUS=logstartFUS,          *
OLDPGM=hac23101PGM,                         *
NEWPGM=PGMCURR,WORK=FUNCPRM
.
.
.
FUNCPRM DS 5F
```

2.10.6 Example 3

Command **VI 101** starts an insertion. The first SCH receives a positive acknowledgment and a VICC is generated. A function entry for this VICC must also be created now.

```

*-----*
*      CALL FGET service to get the current      -
*      PGM entry and the current FUS for the      -
*      VI I01 Function (SCH)                      -
*      Save this FUS in FUSOLD                   -
*      Save this PGM in PGMCURR                  -
*-----*
          ZHCFUNC FUNC=FGET,WORK=FUNCPRM
          L    RF,FUNCPRM+8           get curr active FUS
          ST   RF,FUSOLD
          L    RF,FUNCPRM+16          get curr PGM entry
          ST   RF,PGMCURR

*-----*
*      CALL FCOPY service to create a new FUS      -
*      Entry with all characteristics of an old      -
*      one (FUSOLD) pointed to by FUS= Parameters-
*      Overwrite FUSORIG with "INTERNAL"            -
*      OVERWRITE FUSMSG with the value pointed     -
*      to by HACCCMD parameter                     -
*      Save new found FUS in FUSNEW Field         -
*-----*
          ZHCFUNC FUNC=FCOPY,FUS=FUSOLD,             *
              WORK=FUNCPRM,HACCCMD=CMDFLD
          L    RF,FUNCPRM+8
          ST   RF,FUSNEW

*-----*
*      CALL FSET service to make a FUS specified-
*      in FUS= parm active in a PGM entry        -
*      specified in OLDPGM                      -
*-----*
          ZHCFUNC FUNC=FSET,FUS=FUSNEW,
              OLDPGM=PGMCURR,WORK=FUNCPRM
          .
          .

*-----*
* now put new Dataqueue-entry into SYS/ROB-Queue -
* Here we will concatenate the current FUS with   -
* the DQR via QADFUS and FUSDQR fields           -
*-----*
          L    RF,SU2300
          BALR RE,RF           put into queue

*-----*
*      CALL FSET service again to make the FUS      -
*      from FUSOLD active in PGM entry specified-
*      in OLDPGM                      -
*-----*
          ZHCFUNC FUNC=FSET,FUS=FUSOLD,             *
              OLDPGM=PGMCURR,WORK=FUNCPRM
          FUNCPRM  DS   5F
          FUSOLD   DS   F
          PGMCURR  DS   F
          FUSNEW   DS   F
          CMDFLD   DC   CL8'VICC      '

```

2.11 ZHCGHT

2.11.1 General information

ZHCGHT serves to hold the program release, assembly data/time and program names as constants for programs not starting with ZHCBEG. A trace record can also be created.

Ensure that R5 points to the ICE and R6 to the ICT.

2.11.2 Syntax

ZHCGHT

ZHCGHT must be followed by one or more blanks

MOD=modname

modname = 8 byte character string

Default: spaces

,VERS=haccversion

haccversion = 7 byte character string

Default: VERS=2.4.000

,TRACE=NO

Default: TRACE=NO

,TRACE=YES

2.11.3 Parameters

MOD=modname

Name of the module in which ZHCGHT is used.

VERS=haccversion

Release stand of the module. Enter 2.3.010 for HCC Version 2.3.1 and 2.4.000 for HCC Version 2.4.0.

TRACE=NO / YES

Specifies whether the register contents (TRACE=YES) should be traced as from this point when the module trace has been activated (TRON MOD), or whether register tracing is not required (TRACE=NO).

2.11.4 Example

Module name, Haccversion, Sysdate and Systime are to be passed to a program at the start of the program. Register contents should also be traced when module trace is active.

```
TESTPROG CSECT
    USING *,RF
    ZHCGHT MOD=TESTPROG,VERS=2.4.000,TRACE=YES
    STM    RE,RC,12(RD)
    .
    .
```

2.12 ZHCISP

2.12.1 General information

ZHCISP serves to use the ISPF Service within the ISPF/HCC interface. The ISPLINK program is loaded with LOAD, and the EPA must be saved in a fullword.

2.12.2 Syntax

ZHCISP

ZHCISP must be followed by one or more blanks

PRM=(parm1,parm2,...)

parm1,parm2,... = RX type addresses

,ISP=stor addr

stor addr = RX type address

Default: ISP=ISPPTR

2.12.3 Parameters

PRM=(parm1,parm2,...)

Specifies the parameters in the same sequence as for a direct call to ISPLINK.

ISP=stor addr

Fullword containing the entry point address for ISPLINK.

2.12.4 Example

Display a panel called PANEL in a HCC/ISPF interface program.

```

LOAD EP=ISPLINK
ST R0,ISPPTR
.
.
MVC ISPSERV,=C'DISPLAY' Move ISPF command
MVC ISPVAR1,=C'PANEL' Move panel name
ZHCISP PRM=(ISPSERV,ISPVAR1),ISP=ISPPTR
.
.
ISPPTR DC F'0'
ISPSERV DC CL8'
ISPVAR1 DC CL8'

```

2.13 ZHCMGN

2.13.1 General information

ZHCMGN serves to create the message modules ZHC03000-ZHC03400. It can also be used as mapping macro for the messages when option DSECT=Y is selected.

2.13.2 Syntax

ZHCMGN

ZHCMGN must be followed by one or more blanks

N=number	number = decimal number
,T1='text' ,T2='text' ,T3='text'	'text' = character string
,V1/V2/V3=Ll'l'text' ,V1/V2/V3='text'	ll = decimal number , text = 1 byte placeholder text = character string. Length max. = max. Length of Vx
,H1=N ,H1=Y	Default: H1=N
,H2=N ,H2=Y	Default: H2=N
,H3=N ,H3=Y	Default: H3=N
,DES=N ,DES=Y	Default: DES=N
,SUPR= ,SUPR=Y	Default: SUPR=N
,ACTID=I ,ACTID=A	Default: ACTID=I
,SMF=N ,SMF=Y	Default: SMF=N
,DSECT=N ,DSECT=Y	Default: DSECT=N
,INDEX=char	Char = 1 byte or longer character string Default: INDEX=spaces

2.13.3 Parameters

N=number

Message number, can be 000 - 499 (status 11/93).

T1='text' / T2='text' / T3='text'

Positions of the 1st / 2nd / 3rd constant text portion of the message.

V1=Lll'text' / V2=Lll'text' / V3=Lll'text'**V1='text' / V2='text' / V3='text'**

Positions and lengths of the 1st / 2nd / 3rd variable text portions of the message. V1=Lll'text' specifies the text length explicitly. The text only serves as placeholder for an Assembler DC constant.

For V1='text', the length of 'text' is the implicit length. In this case, ensure that the number of characters in 'text' matches the required length.

H1=N/Y / H2=N/Y / H3=N/Y

Specifies whether the parameters T1 / T2 / T3 are in hexadecimal and must be "unhexed" (Hx=Y) or not (Hx=N).

DES=N/Y

Specifies whether the message should be displayed with Descriptor code 2 (action message) (DES=Y) or not (DES=N).

SUPR=N/Y

Specifies whether the message should be suppressed (SUPR=Y) or not (SUPR=N) when Message Suppression is required (HCC Startparm SUPR=Y) .

SMF=N/Y

Specifies whether the message should be written in the corresponding SYS1.MANx SMF files as SMF record (SMF=Y) or not (SMF=N) .

ACTID=I/A

Action identifier for the message prefix. "I" stands for information message and "A" for action message.

DSECT=N/Y

Specifies whether a mapping DSECT should be created for messages (DSECT=Y) or not (DSECT=N). For DSECT=Y, all parameters apart from INDEX are irrelevant.

INDEX=char

Prefix to be used for the field names in the mapping DSECT.

MACRO DESCRIPTION

2.13.4 Example 1

Include message HAC490A in module ZHC03400. Use descriptor code 2 as action message. Write the record to an SMF file and do not suppress the message.

```
ZHCMGN N=490,T1='Sys ',V1='s',T2=' ',Rob ',   *
      V2='r',T3=' message text ',                 *
      DES=Y,SMF=Y,SUPR=N,ACTID=A
```

2.13.5 Example 2

Message HAC405I in module ZHC03400 serves to output strings already edited by a program. The maximum string length is 100 bytes.

```
ZHCMGN N=405,V1=L100' '
```

2.14 ZHCMMSG

2.14.1 General information

ZHCMMSG serves to output messages within HCC. These messages are passed to the originator of the triggering command.

The macro can be used in re-entrant programs. Use the parameter WORK= in this case. This parameter must point to a 5 fullword area. For action messages, the message WTOID is passed to the calling program in R15 for possible DOMs.

2.14.2 Syntax

ZHCMMSG

ZHCMMSG must be followed by one or more blanks

msgnum	msgnum = decimal number
,var1	var1 = RX type address or Reg. (1) - (15)
,var2	var2 = RX type address or Reg. (1) - (15)
,var3	var3 = RX type address or Reg. (1) - (15)
,BUFFA=stor addr	stor addr = RX type address or (1) - (15)
,WORK=stor addr	stor addr = RX type address

2.14.3 Parameters

msgnum

Positional parameter that must be the first parameter, contains the message number.

var1 / var2 / var3

Pointers to the variable text portions.

BUFFA=storaddr

Pointer to a reply area for the final edited message. This is done in for example, ZHC08100, whereby HACMSGs can be displayed.

WORK=storaddr

Name of a work area that must be within the WORKDSECT for re-entrant programs. This work area must be 5 fullwords long.

MACRO DESCRIPTION

2.14.4 Example

Output a message with variable data in a re-entrant program.

```
XC MSGWORK(20),MSGWORK      Clear ZHCMMSG WORKAREA
MVC SYSTEM,=C'1'            move system
MVC ROBOT,=C'2'            move robot
LA RF,SYSTEM                point to system
LA RE,ROBOT                 point to robot
ZHCMMSG 490,(RE),(RF),WORK=MSGWORK
.
.
.
WORKA DSECT .
SYSTEM DC C' '
ROBOT DC C' '
MSGWORK DS 5F
```

2.15 ZHCSUE

2.15.1 General information

ZHCSUE serves, within HCC, to call the Service subprograms from ZHC06900 (SU....) from a program. Registers 3 and 7-14 of the calling program can also be saved, and trace data can be passed as well.

This macro must not be used for new routines any longer and should be replaced by ZHCBEG TYPE=INT.

2.15.2 Syntax

ZHCSUE

ZHCSUE must be followed by one or more blanks

MOD=modname

modname = max. 8 byte character string

Default: MOD=spaces

,SAVE=YES
,SAVE=NO

Default: SAVE=YES

,TRACE=YES
,TRACE=NO

Default: TRACE=YES

,TRDATA=trdata

trdata = max. 20 byte character string

Default: TRDATA=spaces

2.15.3 Parameters

MOD=modname

Module name appearing as a constant in the object code. The MOD=modname parameter is optional but its use is recommended.

SAVE=YES / NO

Specifies whether registers 7 - 14 of the calling program should be saved (SAVE=YES) or not (SAVE=NO).

TRACE=YES / NO

Specifies whether a macro entry trace record should be created (TRACE=YES) or not (TRACE=NO).

TRDATA=trdata

Specifies whether a max. 20 byte character string should also be output when trace is active (TRACE=YES).

MACRO DESCRIPTION

2.15.4 Example

Start subroutine SU2400 :

```
U2400    CSECT
        USING  *,RF
        ZHCSUE MOD=U2400
        LR      RC,RF
        DROP   RF
        USING U2400,RC
        .
        .
```

2.16 ZHCSUX

2.16.1 General information

ZHCSUX serves, within HCC, to exit ZHC06900 subroutines already started with ZHCSUE. Registers 7-14 can be restored again (set the parameter SAVE=YES in the ZHCSUE macro in this case), and a macro entry trace record can also be created.

This macro must not be used for new routines any longer and should be replaced by ZHCEXIT.

2.16.2 Syntax

ZHCSUX

ZHCSUX must be followed by one or more blanks

OFFSET=offset

offset = decimal number/multiple of 4

Default: OFFSET=0

,SAVE=YES

Default: SAVE=YES

,SAVE=NO

,TRACE=YES

Default: TRACE=YES

,TRACE=NO

2.16.3 Parameters

OFFSET=offset

Offset to be added to the return address in R14. The offset must be a multiple of 4 otherwise unforeseen events can occur in the calling program.

SAVE=YES / NO

Specifies whether registers 7 - 14 of the calling program should be restored (SAVE=YES) or not (SAVE=NO).

TRACE=YES / NO

Specifies whether a macro entry trace record should be created (TRACE=YES) or not (TRACE=NO).

MACRO DESCRIPTION

2.16.4 Example

Call subprogram SUXXXX in a program. Processing should continue after the return to the calling program depending on the result from the subprogram.

Calling program:

```

      L      RF ,SUXXXX
      BASR  RE ,RF
      B     SUBOK
      B     SUBNOK
SUBOK   DS    0H
.
.
.
SUBNOK  DS    0H

```

Called program:

```

UXXXX  CSECT
      USING * ,RF
      ZHCSUE MOD=UXXXX
      LR    RC ,RF
      DROP RF
      USING UXXXX,RF
.
.
.
proceed subroutine function
.
.
.
SUBOK   DS    0H
      ZHCSUX OFFSET=0
SUBNOK   DS    0H
      ZHCSUX OFFSET=4

```

2.17 ZHCTME

2.17.1 General information

ZHCTME serves, within HCC, to convert a binary time and date to

- Julian date (DD.MM.YY), time (HH.MM.SS) and weekday
- AML send timestamp (TT/HHMMSS).

2.17.2 Syntax

ZHCTME

ZHCTME must be followed by one or more blanks

TIME=storaddr

storaddr = RX type address or (0) - (15)
required parameter

,DATE=storaddr

storaddr = RX type address or (0) - (15)
required parameter

,STAMP=storaddr

storaddr = RX type address or (0) - (15)

,JDATE=storaddr

storaddr = RX type address or (0) - (15)

2.17.3 Parameters

TIME=storaddr

Time in binary format. This time can be determined with the TIME BIN macro.

DATE=storaddr

Date in packed format. This date can also be determined with the TIME BIN macro.

STAMP=storaddr

Specifies where the time/date converted to the HCC send format (DD/HHMMSS) should be stored. When this parameter is omitted, the HCC send format is stored in the data area of ZHCTMD (TMDSTMP) (refer to examples).

JDATE=storaddr

Specifies where the time/date converted to a Julian date (DD.MM.YYHH.MM.SStttttttt) should be stored. When this parameter is omitted, the Julian date is stored in the data area of ZHCTMD (TMDRES) (refer to examples).

MACRO DESCRIPTION

2.17.4 Example 1

Determine and convert the date and time in a program, and then enter the HCC send format directly in the corresponding data queue entry.

```

TIME BIN
ZHCTME TIME=( 0 ),DATE=( 1 ),STAMP=QABATIME
.
.
.
ZHCTMD      This macro must be specified

```

2.17.5 Example 2

Determine and convert the date and time in a program, and then output the result.

```

TIME BIN
ZHCTME TIME=( 0 ),DATE=( 1 )
MVC WORK(7),=C'Date: '
MVC WORK+7(8),TMDDATE
MVC WORK+15(11),=C'Time: '
MVC WORK+26(8),TMDTIME
MVC WORK+34(12),=C'Weekday: '
MVC WORK+46(10),TMDWDAY
MVC WORK+56(17),=C'HACC timestamp: '
MVC WORK+73(9),TMPSTMP
ZHCMSG 36,WORK
.
.
.
ZHCTMD      This macro must be specified
WORK DC    CL100'

```

2.18 ZHCTRE

2.18.1 General information

ZHCTRE serves, within HCC, to create **trace points** in programs. Trace data are then created at this trace point when such a trace point has been activated (**SET TRACE,ON,..** command) and the Trace function has been started (**SET TRACE,START** command). Trace data contents depend on the **SET TRACE,ON** command. Three trace levels are available:

- Level 1 is a flow trace. CSECTNAME and an arbitrary name (for example, a label name) are displayed.
- Level 2 displays the contents of all registers.
- Level 3 displays data pointed to by a register.
- **ATTENTION:** ZHCTRE must not be used in ZHC04800 or ZHC00200 because this leads to recursive calls to the respective program.

Set the desired level with the **SET TRACE,ON,LVL=..** command. All levels below the selected level are also activated. When this is not required, deactivate single levels with the **SET TRACE,OFF,LVL=..** command.

The syntax of the **SET TRACE,...** command is described in detail in the HCC DIAGNOSTIC GUIDE.

2.18.2 Syntax

ZHCTRE

ZHCTRE must be followed by one or more blanks

NUM=number	number = decimal number - required parameter
,NAME=charstring	charstring = max. 8 byte character string required parameter
,LEN=length	length = decimal number Default: LEN=32
,WORK=storaddr	storaddr = RX type address

2.18.3 Parameters

NUM=number

Assigns a unique number within the CSECT to the trace point as identification. The syntax supports duplicate numbers but this is not recommended.

NAME=charstring

Names the trace point, for example, a label name.

LEN=length

Trace data length for trace level 3.

WORK=storaddr

Macro work area for re-entrant programs to be used to pass parameters to the trace routine. This work area must be at least 104 bytes long.

MACRO DESCRIPTION

2.18.4 Example

Create a trace point at label XYZ in a re-entrant program. Four other trace points already exist in this CSECT. Sixty bytes of data should be formatted for each register specified when a level 3 trace is active. Create a new trace point at label YYY as well.

```
XYZ      DS    0H
        ZHCTRE NUM=5 ,NAME=LABXYZ ,LEN=60 ,WORK=TRCWORK
        .
        .
YYY      DS    0H
        ZHCTRE NUM=6 ,NAME=LABYYY ,WOR=TRCWORK
        .
        .
WORKA    DSECT
TRCWORK  DS    CL104
```

2.19 ZHCUPE

2.19.1 General information

ZHCUPE serves, within HCC, to provide entry code for arbitrary internal/external CSECTs. ZHCUPX provides the exit code. Some special features must be considered:

- Only one base register may be used.
- Re-entrant programs cannot use ZHCUPE/ZHCUPX at the present time.
- The registers of the calling program are stored in a data area where the name is the CSECT name suffixed by \$X (see example).
- The macro provides a return code field where the name is the CSECT name suffixed by RC. ZHCUPX loads the contents of this field in R15.

2.19.2 Syntax

ZHCUPE

ZHCUPE must be followed by one or more blanks

MOD=modname

modname = max. 6 byte character string
required parameter

,BASE=basenum

basenum = decimal numeric
required parameter
Default: BASE=12

,PRINT=GEN
,PRINT=NOGEN

Default: PRINT=NOGEN

2.19.3 Parameters

MOD=modname

Module name appearing as a constant in the object code.

BASE=basenum

Base register.

PRINT=GEN/NOGEN

Specifies whether the macro should be resolved (PRINT=GEN) or not (PRINT=NOGEN).

2.20 ZHCUPX

2.20.1 General information

See ZHCUPE

2.20.2 Syntax

ZHCUPX

ZHCUPX must be followed by one or more blanks

2.20.3 Example for ZHCUPE/ZHCUPX

Call the internal CSECT XYZ in a program. R1 contains the ICT address as a parameter. Branch according to the contents of R1 after routine execution.

```

XYZ      DS    0H
Calling program:
        L      RF,=V(XYZ)      get EPA of program
        LR    R1,R6
        BALR RE,RF
        B     *+4(RF)
        B     RCGOOD
        B     RCBAD
        B     CATASTR
RCGOOD   DS    0H
RCBAD    DS    0H
CATASTR  DS    0H
.
Called program:
XYZ      CSECT
ZHCUPE  MOD=MODXYZ,BASE=12
LR      R6,R1
USING   HACICE,R6
.
.
.
RCGOOD   DS    0H
        XR    RF,RF      set RC=0
        ST    RF,XYZRC   Csectname + RC
        B     XYZEXIT
RCBAD    DS    0H
        XR    RF,RF      set RC=4
        LA    RF,4
        ST    RF,XYZRC   Csectname + RC
        B     XYZEXIT
CATASTR  DS    0H
        XR    RF,RF      set RC=8
        LA    RF,8
        ST    RF,XYZRC   Csectname + RC
        B     XYZEXIT
XYZEXIT  ZHCUPX

```



MACRO DESCRIPTION